|  |  |
| --- | --- |
| Student Name: | Nathan Strawhand |
| Course: | CIS339 |
| Session (month, year): | November 2017 |

Object-Oriented System Analysis and Design

The School of Prosperity  
Student Records System (SRS)

[Week 1—System Request 3](#_Toc360108365)

[Week 2—Use Case Diagram AND Use Cases Descriptions 5](#_Toc360108366)

[Week 3—Class Diagram AND CRC Cards 9](#_Toc360108367)

[Week 4—Sequence, Communication, and State Machine Diagrams 15](#_Toc360108368)

[Week 5—Package Diagram 19](#_Toc360108369)

[Week 6—Method Contract AND Method Specification 21](#_Toc360108370)

[Week 7—Object-Oriented Application Coding 24](#_Toc360108371)

# Week 1—System Request

|  |  |
| --- | --- |
| 1 | Use this system request template and complete the SRS system request.  **System Request:**  Student Record System  **Project sponsor:**  The School of Prosperity  **Business Need:**  The School of Prosperity is seeking a new way to register students, which currently is labor intensive, error-ridden at times, and expensive. They want to transform the current system into an online system that is accessible to students as well. This new system will allow students to register for classes online, without the need of a school staff member to do so, although they still can have a staff member attend their needs regarding classes. It will also allow staff to maintain student, course, and class records for enrolled students.  **Business Requirements:**  The business requirements would require a very unique system that can allow students to register, add or drop classes, without needing to be face to face with a school employee. This would require logins for students, a database that keeps record of all student activity and information, as well as students understanding the system once a new school session begins. This may require more advanced technology, as well as internet access on the school’s part. As well as keeping the network secure. Ultimately this system will allow students to self-register, allow staff to register students and have maintainable abilities with records, classes and courses.  **Business Value:**  The value that this system will provide lies more in just making registering students easier. It will provide more time for employees to complete other work-related tasks. The current in-house system appears very time consuming, hour intensive, and non-consistent. As well as accident prone, the current system requires hard paper copies of enrolling students, making it expensive as well. This new system appears eco-friendlier as well as less expensive on the SOP. In respect to saving money, the new system is estimated to save $3,000 a year in paper goods, as well as increase productivity for staff by %10. With the expected increase of student enrollment and productivity, it is expected the savings for an entire year to be $50,000.  **Special Issues or Constraints:**  This system would require savvy developers, as well as users. However, one cannot judge every user’s computer literacy levels. The system would also require training on the staff, as well as students. Developing a new system would be expensive and labor intensive as well, without the guarantee it will work perfectly all the time. This system would be power and internet dependent, requiring the SOP to update network services, as well as possibly hiring network security staff. This system is expected to raise the attendance levels of the school as well, can the school currently accommodate more students? The time frame for the system to be in fully integrated in desktop form is one year, hopefully a goal that can be obtained. |
| 2 | This will be a brand-new system and will require patience. The flow of the system includes a login for each of the users, student and staff and once logged in, the flow afterword will be vastly different. If a student logs in, there will be a different set of pages, rather than a staff member logging in, which will require them to verify student account information. The literacy levels may need to be adjusted for our student users, as well as our staff. With more people on the network, a network administrator may need to be hired, to look out and make sure our school network is secure. With more students enrolling, it may require hiring of more teachers. |
| 3 | I thought through the whole scenario; thinking about all the current problems the school has, while arriving to conclusions of how to remedy them. Traversing a new school registering program may not come easy for some users, however we do live in an advanced technology age. With everything being more online, I knew this would jeopardize the schools network; more users mean more risk of infiltration. Five Questions regarding New System   1. When is the new system expected to be fully integrated and staff adapted to its workings? 2. Would pay increase, regarding staff finishing more other work-related tasks? 3. Should students have to all submit to online logins? 4. Could some stick with “the old-fashioned” way of seeing a staff member for classes? 5. How much would this cost the school? |

# Week 2—Use Case Diagram and Use Cases Descriptions

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Generate a use case diagram for all of your use cases, including their actors. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  **SRS Use Case Diagram**    Use this use case description template and complete the two SRS use cases of (1) Maintain Class Records and (2) Register a Student for Classes.  Maintain Class Records   |  |  |  |  | | --- | --- | --- | --- | | Use Case Name:  Maintain Class Records | | ID:  100 | Importance Level:  Very High | | Primary Actor:  Staff | Use Case Type:  Detail and Essential | | | | Stakeholders and Interests:  Staff and Student | | | | | Brief Description: | | | | | Trigger: The need to update, or maintain the students, courses and classes.  Type: External | | | | | Relationships:  Association: Staff members  Include: Login  Extend:  Generalization: | | | | | Normal Flow of Events:   1. A staff member logs in to the system. The staff member enters the student ID, then selects the course/s they want to add, delete, or update. A list of all the classes for that course is then displayed, they chose the class they want to add, delete or update. If they are adding a class to a student schedule, they select the begin date for the class and a location. If the class is online, it will display URL. The system will validate any class questions, and enrollment will take place if validation passes. | | | | | SubFlows:  Validation is required for a student to take or add a class. This is checking class availability, if the student is already taking or registered for the class, meets GPA requirement, and if they completed any pre-requisite classes. | | | | | Alternate/Exceptional Flows:  When a course or student I.D. doesn’t exist or isn’t found, an error message would appear, asking to re  enter information. | | | |   **Register a Student for Classes**   |  |  |  |  | | --- | --- | --- | --- | | Use Case Name:  Register Student for Classes | | ID:  101 | Importance Level:  High | | Primary Actor:  Students, Staff | Use Case Type:  Detail and Essential | | | | Stakeholders and Interests:  Students may self-register when pre-reques are validated. Staff may also register students for  classes. Only staff members can maintain class, student, and course records. | | | | | Brief Description:  Students can register for classes without needing to physically meet with a staff member.  Students also have the option to register with a staff member on campus. The SRS will allow  access from any computer provided it is connected to the internet. | | | | | Trigger: Once a student registers, they will then be able to start choosing courses to take. Staff  Can register students as well to take course.  Type: External | | | | | Relationships: Association: Students can register for classes, as well as staff can register students to take classes.  Include: Login to SRS  Extend:  Generalization: | | | | | Normal Flow of Events:  A staff member or student logs into the system. A list of available courses offered is displayed. If a staff member logs in, they will be asked for a student ID. A list of classes will be displayed based on the student’s degree path. A class is selected. If student meets the requirements to take the course, then the student can add, modify or delete. If not, an eligibility message will appear. Once the student is registered, a confirmation will appear. | | | | | SubFlows:  Validating registration includes student eligibility based on, if the course has room, is being offered during the desired semester, if there is duplicate registration, any GPA requirements, and if prerequisites courses are met. Another aspect that is checked is if the student is already taking the maximum number of classes. | | | | | Alternate/Exceptional Flows:  If the student isn’t found through their ID as well as course or classes or if they aren’t able to  take the class an error message should appear. | | | | |
| 2 | Validate and verify your use case diagram and use case descriptions against the SRS Requirement Definition and the SRS System Request. Students and staff both can register the student for classes. Successful registration depends on validating that all criteria and prerequisites are met, as validation is required for a student to be enrolled in the class. Staff members are also able to maintain class records, however students may not. |
| 3 | Explain how you completed your work, the decisions you made to arrive at your conclusions, and the lessons you learned.  I created a Use Case Diagram to show how the actors would use the student registration system. Students don’t get as many privileges as the staff; however, they can register for courses. Based on the criteria given I determined the different use cases that were allowed, per actor. I knew that the validation would be a sub flow since there are special needs that must be obtained in order for a student to take that class. In addition, I know that the students and staff will have to both login to the system, however each login will be vastly different. |

# Week 3—Class Diagram and CRC Cards

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Generate a class diagram for the SRS system. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  **SRS Class Diagram**    Use this CRC template and complete a CRC card for each class you designed in your SRS class diagram.   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Generate a class diagram for the SRS system. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  **SRS Class Diagram**    Use this CRC template and complete a CRC card for each class you designed in your SRS class diagram.  **Class1 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  Student | **ID:**  1 | | **Type:** | | **Description:**  Maintains information about the student as well as registration | | | **Associated Use Cases:**  Maintains Students Registration  Maintains Student Records | | **Responsibilities**  Create Student  GetStudent by StudentID  Update Student Information  Delete Student | | **Collaborators**  Registration | | | **Back:** | | | | | **Attributes:**  StudentID: int  FirstName: string  MiddleInitial: string  LastName: string  Department: string  DateOfBirth: date  GPA: double  OnlineEquipment: Boolean | | | | | **Relationships:**  **Generalization (a-kind-of):**  **Aggregation (has-parts):**  **Other Associations:**  Registration | | | |   **Class2 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  Registration | **ID**  2 | | **Type:** | | **Description:**  Maintains relationships that exist between classes and students. Each instance of this class is a registration record of a student registered for a class. It maintains information of a student registered for classes. | | | **Associated Use Cases:**  Register Student for Classes | | **Responsibilities**  CreateRegistrationRecord  UpdateRegistrationRecord  DeleteRegistrationRecord  GetRegistrationRecord  ProcessFinalGrade  VerifyRegistrationRules  RegisterStudent  DisplayConfirmationMessage  DisplayViolationMessage | | **Collaborators**  Student, Class | | | **Back:** | | | | | **Attributes:**  StudentID  ClassID  DateRegistered: date  StudentStatus: char (R = Registered, A = Active, D = Dropped, C = Completed Class)  Grade: char (A = Excellent, B = Good, C = Fair, D = Poor, F = Fail) | | | | | **Relationships:**  **Generalization (a-kind-of):**  **Aggregation (has-parts):**  **Other Associations:**  Student, Class | | | |   **Class3 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  Class | **ID:**  3 | | **Type:** | | **Description:**  This class maintains the attributes common to all kinds of classes like CourseID, BeginDate, EndDate. It also has methods maintaining the classes and Get class information. This class is associated with Online and Face to Face subclasses. | | | **Associated Use Cases:**  Maintain Class Records  Register Student for Classes | | **Responsibilities**  CreateClass  UpdateClass  DeleteClass  GetClass  GetCourseClassList  UpdateClassEnrollment | | **Collaborators**  Registration  Course | | | **Back:** | | | | | **Attributes:**  CourseID: string  BeginDate: date  EndDate: date  MaximumEnrollment: integer  ClassCount: integer  ClassStatus: string (X= Cancelled, A=Active, C=Completed) | | | | | **Relationships:**  **Generalization (a-kind-of):**  **Aggregation (has-parts):**  **Other Associations:**  Course, Registration | | | |   **Class4 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  Course | **ID:**  4 | | **Type:** | | **Description:**  This class named Course maintains information about a Course | | | **Associated Use Cases:**  Maintain Course Records  Register Student for Classes | | **Responsibilities**  CreateCourse  UpdateCourse  DeleteCourse  GetCourseByCourseID  GetAllAvailableCourses | | **Collaborators**  Class | | | **Back:** | | | | | **Attributes:**  CourseID: string  CourseName: string  CreditHours: int  Description: string  PrerequisiteCourse: string  AvailabilityStatus: string | | | | | **Relationships:**  **Generalization (a-kind-of):**  **Aggregation (has-parts):**  **Other Associations:** Class | | | |   **Class5 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  FaceToFace | **ID:** | | **Type:** | | **Description:**  Has information regarding a class if its face to face, on campus | | | **Associated Use Cases:**  Maintain Class Records  Register Student for Classes | | **Responsibilities**  CreateFaceToFaceClass  UpdateFaceToFaceClass  DeleteFaceToFaceClass  GetFaceToFaceClass | | **Collaborators**  Course  Registration | | | **Back:** | | | | | **Attributes:**  Building: string  room: string | | | | | **Relationships:**  **Generalization (a-kind-of):**  Class  **Aggregation (has-parts):**  **Other Associations:**  Register, Course | | | |   **Class6 CRC Card**   |  |  |  |  | | --- | --- | --- | --- | | **Front:** | | | | | **Class Name:**  Online | **ID:** | | **Type:** | | **Description:**  Has information regarding an online class, not an on-campus class | | | **Associated Use Cases:**  Maintain Class Records  Register Student for Classes | | **Responsibilities**  CreateOnlineClass  DeleteOnlineClass  UpdateOnlineClass  GetOnlineClass | | **Collaborators**  Course  Registration | | | **Back:** | | | | | **Attributes:**  URL: string  Browser: string | | | | | **Relationships:**  **Generalization (a-kind-of):**  Class  **Aggregation (has-parts):**  **Other Associations:**  Registration, Course | | | | | |
| 2 | Validate and verify your class diagram and CRC cards against the SRS use case diagram and use case descriptions. The CRC cards show each of the classes, including attributes and associations within other classes in my system. We can see that there are 6 different classes, student, Registration, Course, Class, Online, and FaceToFace. They all have their own attributes as well as behaviors. |
| 3 | Explain how you completed your work, the decisions you made to arrive at your conclusions, and the lessons you learned. I decided to make 6 different classes. In order show how the system will work, I included the attributes as well as the behaviors for each class. Some classes have relationships with others, and some classes have aggregations to some classes. For example, Online and FaceToFace are types of classes, and they can live on without the Class class. For the attributes I needed to figure out what makes the class unique, or what the class is “made up of”. Some attributes can be used in multiple classes. |

# Week 4—Sequence, Communication, and State Machine Diagrams

|  |  |  |
| --- | --- | --- |
| 1 | Generate a sequence diagram for the SRS system Register a Student for Classes use case. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  Sequence Diagram for the Register a Student for Classes Use Case     |  | | --- | | Generate a communication diagram for the SRS system Register a Student for Classes use case. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  Communication Diagram for the Register a Student for Classes Use Case |   Generate a state machine diagram for the SRS system Registration Record object/class (the class that maintains the registration of a student in a class). The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  State Machine Diagram for the Registration Record Object |
| 2 | For a student to register for classes, a few steps must be taken. They will first log in. Then they select a course by using one of 2 possible methods. They can select a course with either the course ID or the course name. They will then select a class from the available options listed from the course information. After a class is selected from a list, they then determine whether the class is taken online or on campus. A series of pre-requisites must be obtained before a student can select a class, and then be registered. |
| 3 | This lab was a little harder for me, although I feel as though I have a handle on how the messages and lifelines are created within each diagram. From a student selecting a course, to them selecting a class, the student must know what they are looking for. Otherwise some error messages are generated, depending on the error committed. Once a course is selected, then a list of classes will appear. The student then choses a class. The ability to register for a class depends on if the student meets the needed pre-requisites |

# 

# Week 5—Package Diagram

|  |  |
| --- | --- |
| 1 | Generate a package diagram for the SRS system. The diagram must be generated by a UML drawing tool. Copy and paste your diagram here:  **SRS Package Diagram** |
| 2 | Validate and verify your package diagrams against the SRS class diagram and the SRS Register a Student for Classes use case sequence and communication diagrams.  Packaging the classes not only looks more professional, it partitions what could be groups based on relationships as well as class behaviors. With the data that we are working with Student and Registration is each of their own packaging. Course then is packaged, with similar classes within including Class, FaceToFace, and Online. |
| 3 | Explain how you completed your work, the decisions you made to arrive at your conclusions, and the lessons you learned. I came to the conclusions by partitioning the classes to how they relate and associate with each other. Classes are also packaged by behaviors. Course contains Class, FaceToFace, Online, which all relate somehow. Class is a form of Course, by having similar attributes and behaviors. FaceToFace and Online are aggregates of Class; they are less generalized and more specific. Student and Registration are their own packaging since they do not relate closely to each other or to the other classes in our system. |

# Week 6—Method Contract and Method Specification

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Use this method contract template and complete a method contract for the GetCourseByCourseID() method of the CourseList class. Contracts TEMPLATE  |  |  |  | | --- | --- | --- | | Method Name: GetCourseByCourseID | Class Name: Course | ID: | | Clients (Consumers):  Students  Staff | | | | Associated Use Cases:  Maintain Course Records  Register Student For Classes | | | | Description of Responsibilities:  CreateCourse  UpdateCourse  DeleteCourse  GetCourseByCourseID  GetAllAvailableCourses | | | | Arguments Received: | | | | Type of Value Returned:  CourseID , CourseName, ClassID, ClassName | | | | Pre-Conditions:   1. Validate courseID, return course or list of courses/ and classes to register for | | | | Post-Conditions:   1. Error if wrong course ID was entered, or if course doesn’t exist. | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Use this method specification template and complete a method specification for the GetCourseByCourseID() method of the CourseList class. Method Specification TEMPLATE  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Method Name:GetCourseByCourseID | | | | Class Name: Course | | | ID: | | Contract ID: | | | | Programmer: Michelle | | | Date Due: | | Programming Language:  Visual Basic Smalltalk C++ Java | | | | | | | | | Triggers/Events:  Register Student for Course  Search for Classes | | | | | | | | | Arguments Received:  Data Type: | | Notes: | | | | | | CourseID string | |  | | | | | | CourseName string | |  | | | | | |  | |  | | | | | |  | |  | | | | | | Messages Sent & Arguments Passed:  ClassName.MethodName: | | | | Data Type: | Notes: | | | CourseID | | | | string |  | | | CourseName  Description | | | | String  string |  | | | PrerequisiteCourse | | | | string |  | | | CreditHours | | | | int |  | | | AvailabilityStatus | | | | string |  | | |  | | | |  |  | | |  | | | |  |  | | |  | | | |  |  | | | Argument Returned:  Data Type: | | Notes: | | | | | | CourseName string  ClassName  ClassID | |  | | | | | | Algorithm Specification: | | | | | | | | |
| 2 | Validate and verify your method contract and method specification against the CRC card for the CourseList class and the CRC card for the Course class.  GetCourseByCourseID reflects what is being asked, and returning the value. Entering course ID will return a course list of classes available to be registered for. |
| 3 | Explain how you completed your work, the decisions you made to arrive at your conclusions, and the lessons you learned. The values that are returned include a course list of classes available to take. The GetCourseByCourseID is triggered when a student or staff is looking to register a student or maintain class records. Once they search for the courseID, a list will show up, or an error message depending on the accurate courseID entered. |

# Week 7—Object-Oriented Application Coding

|  |  |
| --- | --- |
| 1 | Copy your code text of the CourseList.GetCourseByCourseID() method and paste it here:  **Code Text of Your CourseList.GetCourseByCourseID() Method**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace CIS339\_Week7\_Lab  {  class Course  {  // property of Class  public string CourseID;  public string CourseTitle;  public int CreditHours;  public string Description;  public string PrerequisiteCourse;    // Class costructor  public Course(string id, string title, int creditHours, string description, string prerequisiteCourse)  {  this.CourseID = id;  this.CourseTitle = title;  this.CreditHours = creditHours;  this.Description = description;  this.PrerequisiteCourse = prerequisiteCourse;  }  }  }  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CIS339\_Week7\_Lab  {  class CourseList  {  public Course[] CourseArray =  {  new Course ("CIS 400", "OO Analysis & Design", 4, "Important class", "CIS 110") ,  new Course ("CIS 150A" , "VB.NET Programming", 4, "Good Introduction to programming", "CIS 100") ,  new Course ("CIS 150B", "C# Programming with labs", 4, "Follow-up to CIS 100", "CIS 100")  };  public Course GetCourseByCourseID(string id)  {  foreach (Course course in CourseArray)  if (course.CourseID == id)  {  return course;  }  return null;  }  }  }  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace CIS339\_Week7\_Lab  {  class CourseListTest  {  static void Main(string[] args)  {  System.Console.WriteLine("GetCourseByCourseIDTestWhenCourseExists");  GetCourseByCourseIDTestWhenCourseExists();  System.Console.WriteLine("GetCourseByCourseIDTestWhenCourseDoesNotExist");  GetCourseByCourseIDTestWhenCourseDoesNotExist();  Console.ReadLine();  }  public static void GetCourseByCourseIDTestWhenCourseExists()  {  CourseList myCourseList = new CourseList();  Console.WriteLine("Please enter Course ID");  string \_CIS\_400 = Console.ReadLine();  Course myCourse = myCourseList.GetCourseByCourseID(\_CIS\_400);  if (myCourse != null && myCourse.CourseID == \_CIS\_400)  {  System.Console.WriteLine("SUCCESS - GetCourseByCourseIDTestWhenCourseExists(): Returned CourseID equal (" + \_CIS\_400 + ")");  }  else  {  System.Console.WriteLine("ERROR - GetCourseByCourseIDTestWhenCourseExists(): Returned CourseID Not equal (" + \_CIS\_400 + ")");  }  }  public static void GetCourseByCourseIDTestWhenCourseDoesNotExist()  {  CourseList myCourseList = new CourseList();  Console.WriteLine("Please enter Course ID");  string \_CIS\_101 = Console.ReadLine();  Course myCourse = myCourseList.GetCourseByCourseID(\_CIS\_101);//CIS 101  if (myCourse != null)  {  System.Console.WriteLine("ERROR - GetCourseByCourseIDTestWhenCourseDoesNotExist(): should have returned null");  }  else  {  System.Console.WriteLine("SUCCESS - GetCourseByCourseIDTestWhenCourseDoesNotExist(): should have returned null");  }  }  }  }  Copy a screenshot of running the unit test of the CourseList.GetCourseByCourseID() method and paste it here:  **Screenshot of running the Unit Test of CourseList.GetCourseByCourseID() Method**    Zip all the files you used in this coding project; copy and paste the \*.zip file here so that it can be unzipped and loaded into the IDE and run on another computer:  **A \*.zip File of all of the Coding Project Files**  <CIS339_Week7_Lab.zip> |
| 2 | Validate and verify your code and screenshot of the CourseList.GetCourseByCourseID() method against the method contract and the method specification of the same method. |
| 3 | Explain how you completed your work, the decisions you made to arrive at your conclusions, and the lessons you learned. I completed my work in class with professor and class. We used the Course, CourseList CRC cards, and the method contract with specification for the GetCourseByCourseID() method. Then I added the source code based on those documents The lesson I learned is implementation is taking a design model and turning it into executable code successfully. |